gbDB version 0.00

Gyanendra Banjare

- ## What is it?

It is an underdevelopment simple DSMS written in python3. It majorly uses SQL along with some exclusive commands, as it is not completely developed SQL is not completely supported and some syntax is a little different than the standard SQL. It can perform simple operations like creating, deleting databases or tables, viewing data, etc.(further described in sections below). Apologies if the program crashes.

- ## How does it store data?

A main working folder "database" is created in the parent directory of "lib" folder ("lib" folder contains the main scripts). You can change the location of the working folder by overwriting the path in "config.config" file. Databases created are nothing but Folders of that name inside our working folder. Tables created inside a database are folders, inside them are two files .yml and .dat file. .yml file contains information about the table names and their data_types(and attributes of data_types if any). The .dat file contains the data entered in the form of comma separated values.

- ## How to run the application?

Simply run the 'main.py' script in the 'lib' folder. On the first time prompt_toolkit 3.0 will be installed, and the main working folder along with the config file will be created. You should be able to use it from the second run onwards.

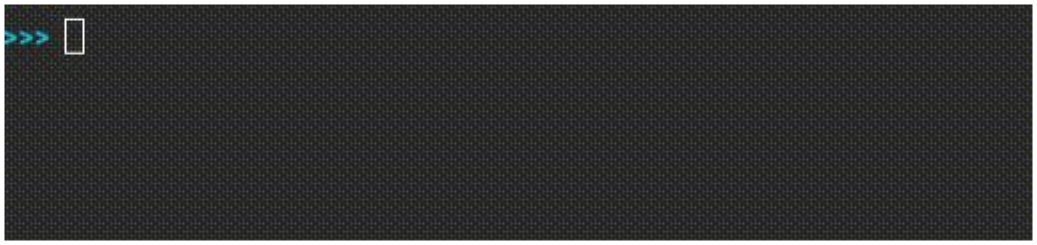
- ## What are the Data_Types available? Is integrity constraint present?

All the data_types except floating point (decimal) values are supported. Integer data_type doesn't take any arguments. Enum and Set are defined as enum('sample_text1' | 'sample_text2' | ...) and set('sample_text1' | 'sample_text2' | ...) i.e using pipe(|) instead of using comma(,) as argument separator. Other data_types have the same definition as in mySQL, with the exception of blob_data_types and string_data_types having the same definitions. YES, integrity check is present just as in mySQL.

- ## How is the interface designed? How does it function?

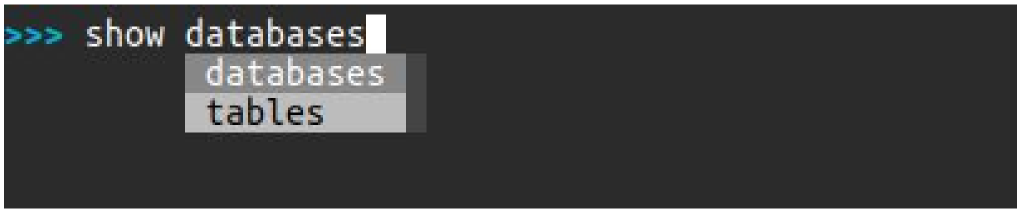
Interface uses prompt_toolkit 3.0 (details). There may be some issues with windows. By default the package installs prompt_toolkit 3.0 on first run, if you are facing issues then you can manually install it. Its functions are as below:

  - ### Full cursor control and history(as in mySQL):

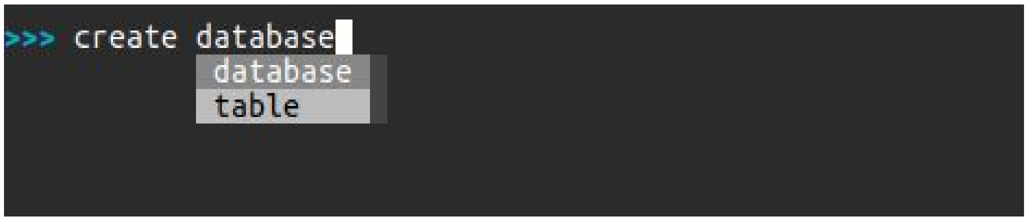


```
>>> 
```

- Auto Suggestion:

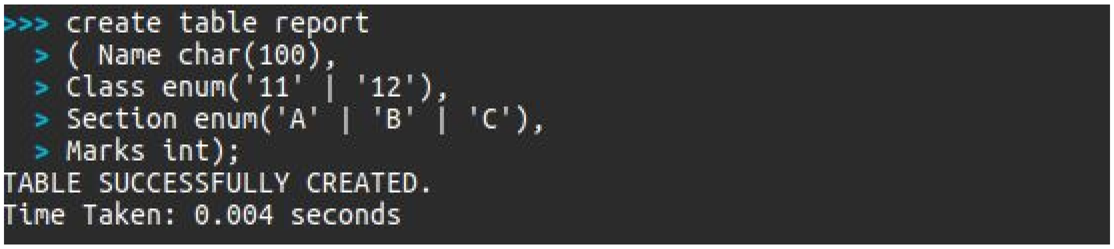


```
>>> show databases
         databases
         tables
```



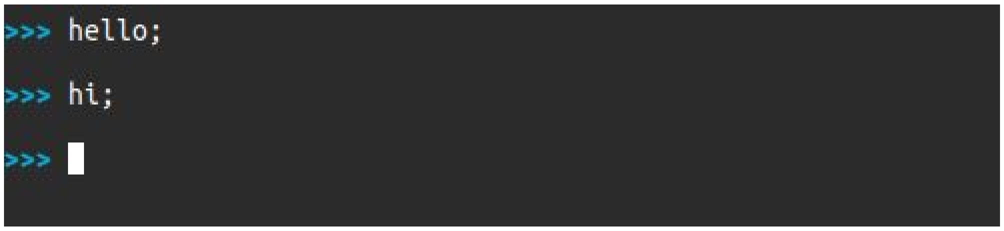
```
>>> create database
           database
           table
```

- Multiple line support:



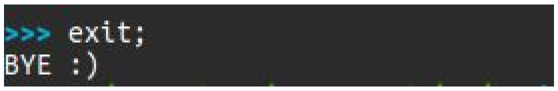
```
>>> create table report
  > ( Name char(100),
  > Class enum('11' | '12'),
  > Section enum('A' | 'B' | 'C'),
  > Marks int);
TABLE SUCCESSFULLY CREATED.
Time Taken: 0.004 seconds
```

- Ignores command if it is not in the list of developed commands(check below):

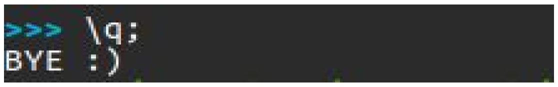


```
>>> hello;

>>> hi;

>>> 
```

- Exiting from application:



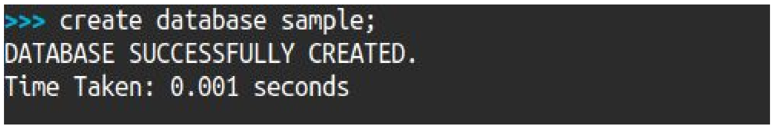
```
>>> exit;
BYE :)
```



```
>>> \q;
BYE :)
```

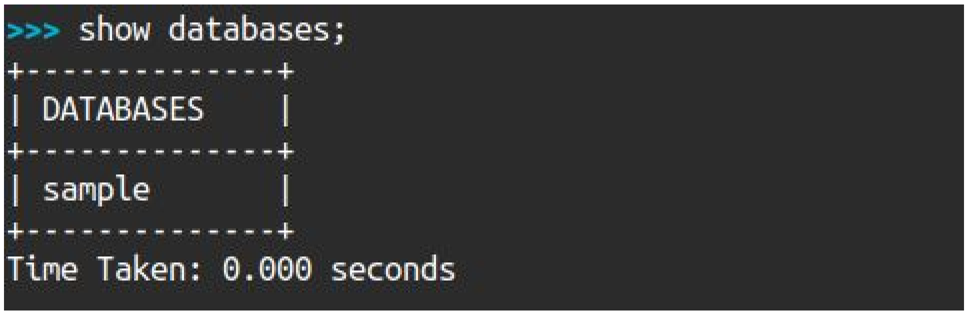
Database commands:

1. create database



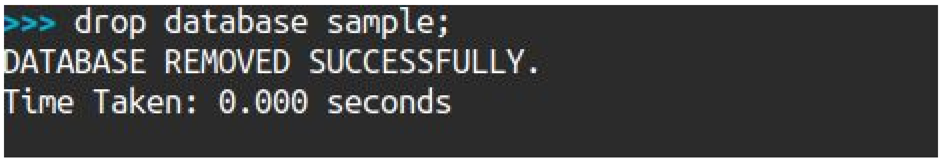
```
>>> create database sample;
DATABASE SUCCESSFULLY CREATED.
Time Taken: 0.001 seconds
```

2. show databases



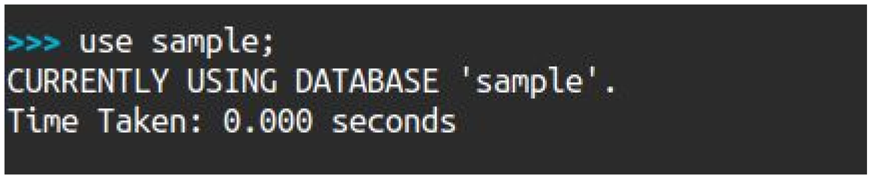
```
>>> show databases;
+--------------+
| DATABASES    |
+--------------+
| sample       |
+--------------+
Time Taken: 0.000 seconds
```

3. drop database:



```
>>> drop database sample;
DATABASE REMOVED SUCCESSFULLY.
Time Taken: 0.000 seconds
```

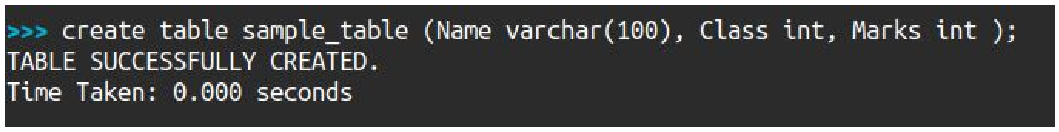
4. Use <database>:



```
>>> use sample;
CURRENTLY USING DATABASE 'sample'.
Time Taken: 0.000 seconds
```

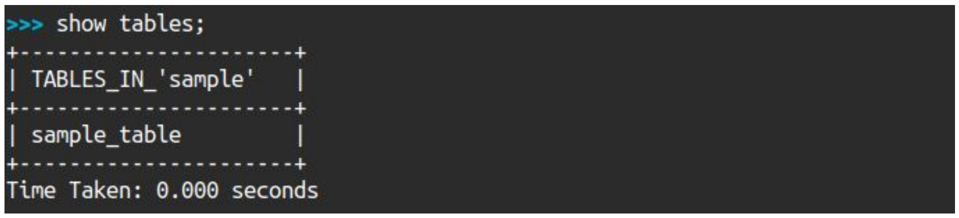
Table commands:

1. Create table:



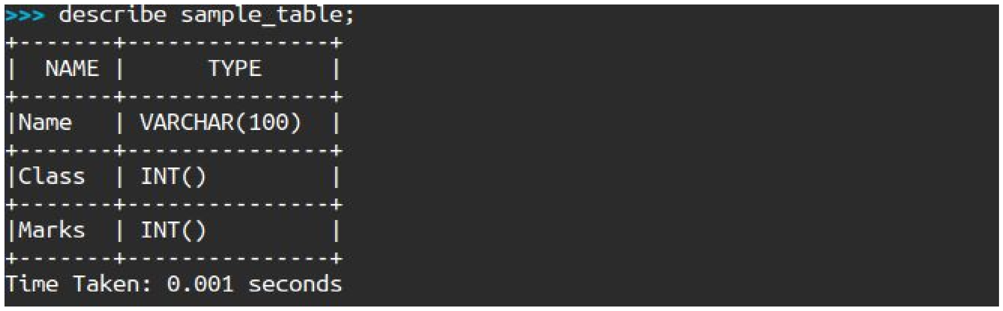
```
>>> create table sample_table (Name varchar(100), Class int, Marks int );
TABLE SUCCESSFULLY CREATED.
Time Taken: 0.000 seconds
```

2. Show tables:



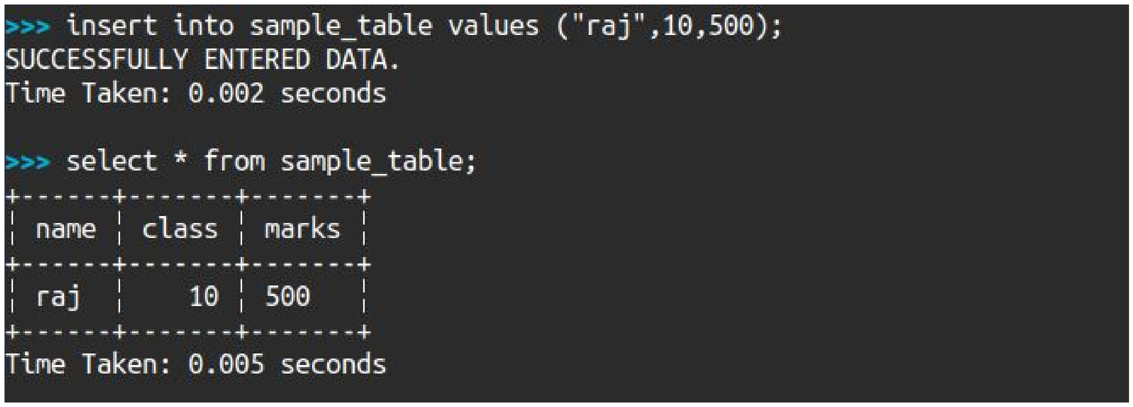
```
>>> show tables;
+---------------------+
| TABLES_IN_'sample'  |
+---------------------+
| sample_table        |
+---------------------+
Time Taken: 0.000 seconds
```

3. Describe <table>:



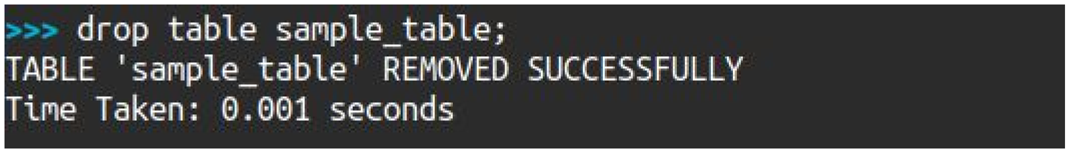
```
>>> describe sample_table;
+-------+--------------+
|  NAME |      TYPE    |
+-------+--------------+
|Name   | VARCHAR(100) |
+-------+--------------+
|Class  | INT()        |
+-------+--------------+
|Marks  | INT()        |
+-------+--------------+
Time Taken: 0.001 seconds
```

4. Insert into <table> values:



```
>>> insert into sample_table values ("raj",10,500);
SUCCESSFULLY ENTERED DATA.
Time Taken: 0.002 seconds

>>> select * from sample_table;
+------+-------+-------+
¦ name ¦ class ¦ marks ¦
+------+-------+-------+
¦ raj  ¦    10 ¦ 500   ¦
+------+-------+-------+
Time Taken: 0.005 seconds
```

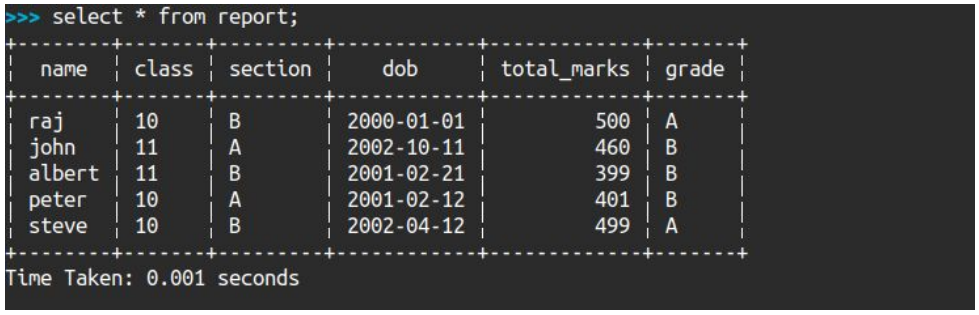
5. Drop table:



```
>>> drop table sample_table;
TABLE 'sample_table' REMOVED SUCCESSFULLY
Time Taken: 0.001 seconds
```

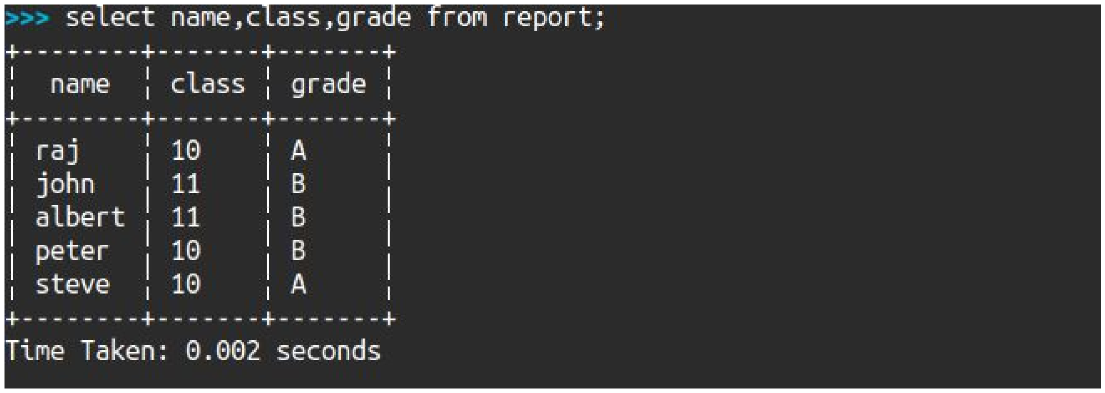
- ## Queries:

1. Select * from <table>:



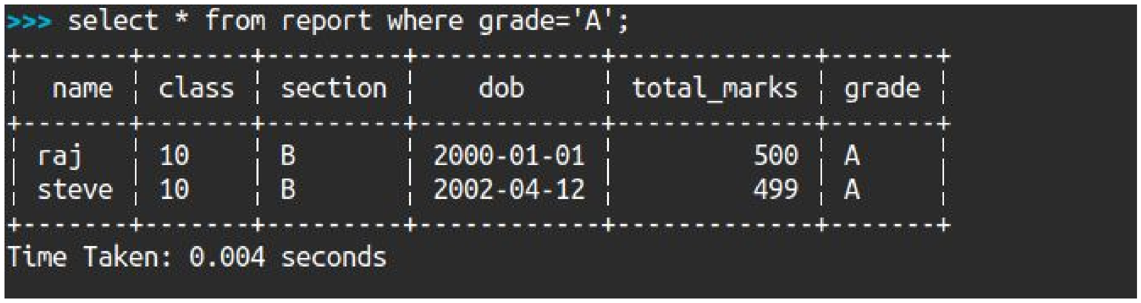
```
>>> select * from report;
+--------+-------+---------+------------+------------+-------+
¦  name  ¦ class ¦ section ¦    dob     ¦ total_marks ¦ grade ¦
+--------+-------+---------+------------+------------+-------+
¦ raj    ¦ 10    ¦ B       ¦ 2000-01-01 ¦        500 ¦ A     ¦
¦ john   ¦ 11    ¦ A       ¦ 2002-10-11 ¦        460 ¦ B     ¦
¦ albert ¦ 11    ¦ B       ¦ 2001-02-21 ¦        399 ¦ B     ¦
¦ peter  ¦ 10    ¦ A       ¦ 2001-02-12 ¦        401 ¦ B     ¦
¦ steve  ¦ 10    ¦ B       ¦ 2002-04-12 ¦        499 ¦ A     ¦
+--------+-------+---------+------------+------------+-------+
Time Taken: 0.001 seconds
```

2. Select column1,column2... from <table>:

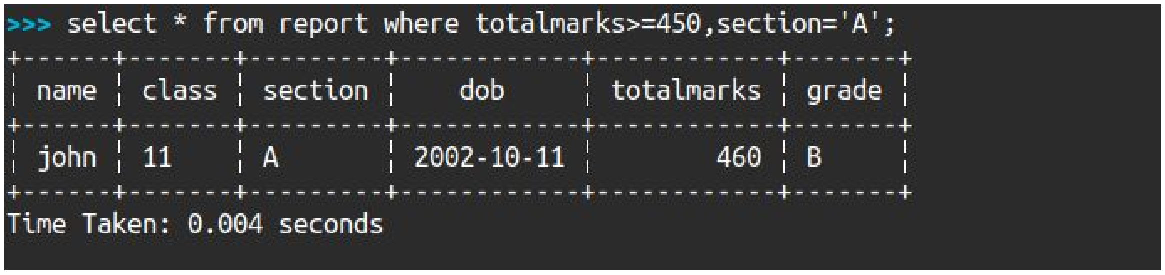


```
>>> select name,class,grade from report;
+--------+-------+-------+
|  name  | class | grade |
+--------+-------+-------+
| raj    | 10    | A     |
| john   | 11    | B     |
| albert | 11    | B     |
| peter  | 10    | B     |
| steve  | 10    | A     |
+--------+-------+-------+
Time Taken: 0.002 seconds
```

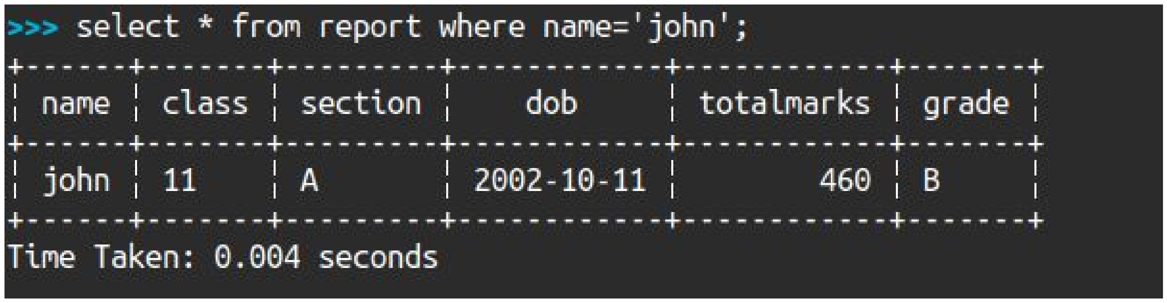
3. Select with where clause(many bugs here):



```
>>> select * from report where grade='A';
+-------+-------+---------+------------+-------------+-------+
|  name | class | section |    dob     | total_marks | grade |
+-------+-------+---------+------------+-------------+-------+
| raj   | 10    | B       | 2000-01-01 |         500 | A     |
| steve | 10    | B       | 2002-04-12 |         499 | A     |
+-------+-------+---------+------------+-------------+-------+
Time Taken: 0.004 seconds
```

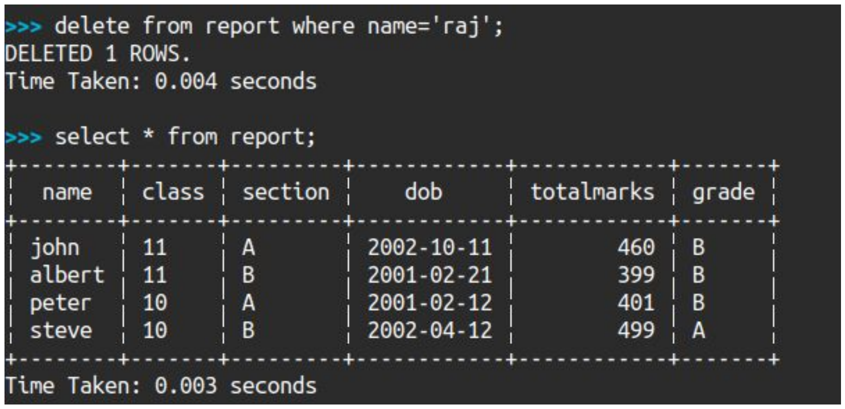


```
>>> select * from report where totalmarks>=450,section='A';
+------+-------+---------+------------+------------+-------+
| name | class | section |    dob     | totalmarks | grade |
+------+-------+---------+------------+------------+-------+
| john | 11    | A       | 2002-10-11 |        460 | B     |
+------+-------+---------+------------+------------+-------+
Time Taken: 0.004 seconds
```



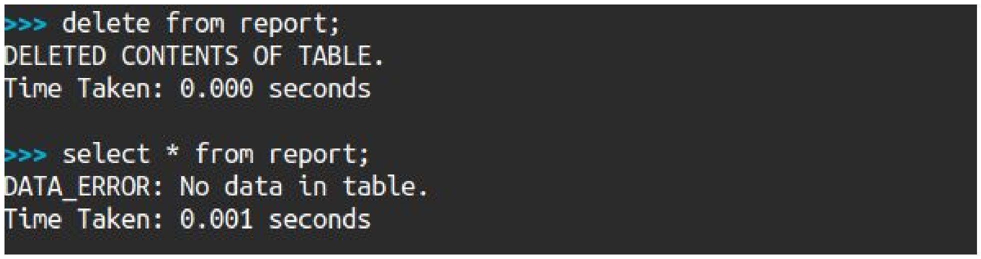
```
>>> select * from report where name='john';
+------+-------+---------+------------+------------+-------+
| name | class | section |    dob     | totalmarks | grade |
+------+-------+---------+------------+------------+-------+
| john | 11    | A       | 2002-10-11 |        460 | B     |
+------+-------+---------+------------+------------+-------+
Time Taken: 0.004 seconds
```

4. Delete from <table> where ... :



```
>>> delete from report where name='raj';
DELETED 1 ROWS.
Time Taken: 0.004 seconds

>>> select * from report;
+--------+-------+---------+------------+------------+-------+
|  name  | class | section |    dob     | totalmarks | grade |
+--------+-------+---------+------------+------------+-------+
| john   | 11    | A       | 2002-10-11 |        460 | B     |
| albert | 11    | B       | 2001-02-21 |        399 | B     |
| peter  | 10    | A       | 2001-02-12 |        401 | B     |
| steve  | 10    | B       | 2002-04-12 |        499 | A     |
+--------+-------+---------+------------+------------+-------+
Time Taken: 0.003 seconds
```

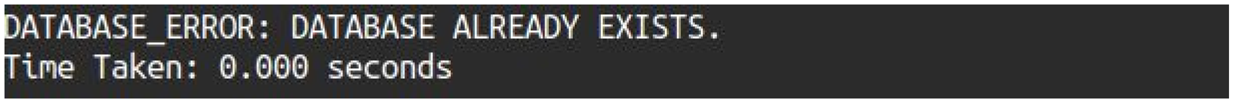
5. Delete from <table>:



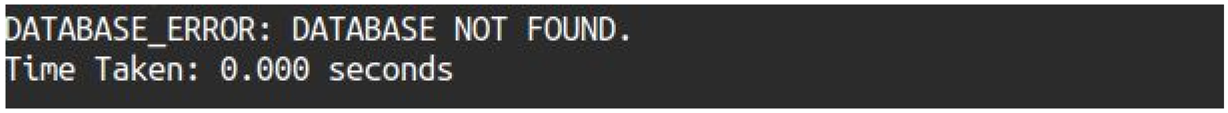
```
>>> delete from report;
DELETED CONTENTS OF TABLE.
Time Taken: 0.000 seconds

>>> select * from report;
DATA_ERROR: No data in table.
Time Taken: 0.001 seconds
```

- ## ERRORS:

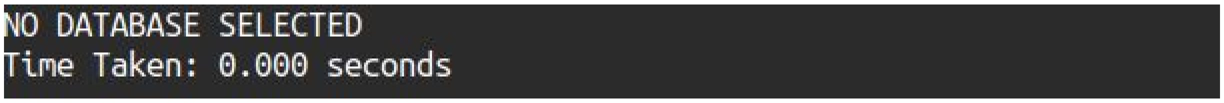
1. ### Database errors:



```
DATABASE_ERROR: DATABASE ALREADY EXISTS.
Time Taken: 0.000 seconds
```

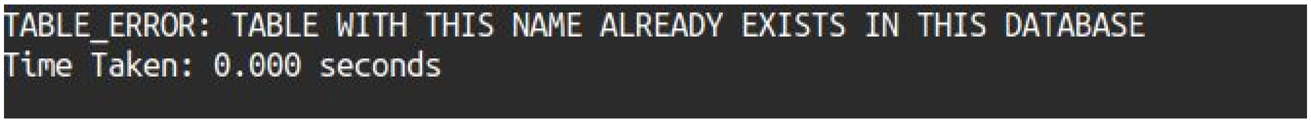


```
DATABASE_ERROR: DATABASE NOT FOUND.
Time Taken: 0.000 seconds
```

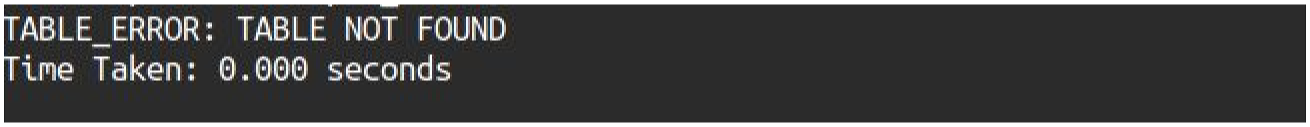


```
NO DATABASE SELECTED
Time Taken: 0.000 seconds
```

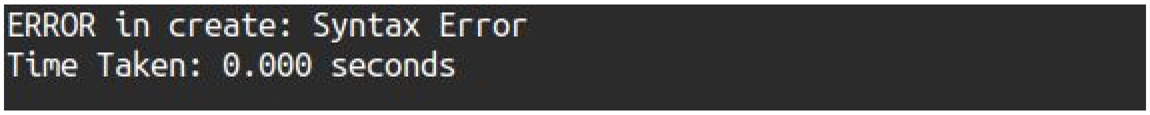
2. ### Table errors:



```
TABLE_ERROR: TABLE WITH THIS NAME ALREADY EXISTS IN THIS DATABASE
Time Taken: 0.000 seconds
```

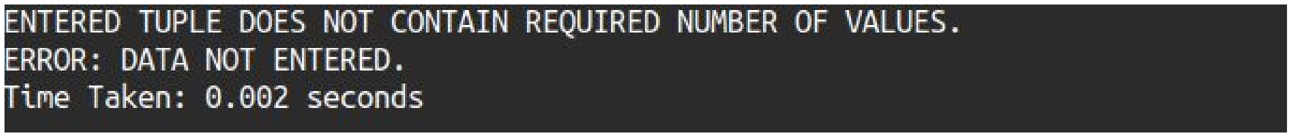


```
TABLE_ERROR: TABLE NOT FOUND
Time Taken: 0.000 seconds
```

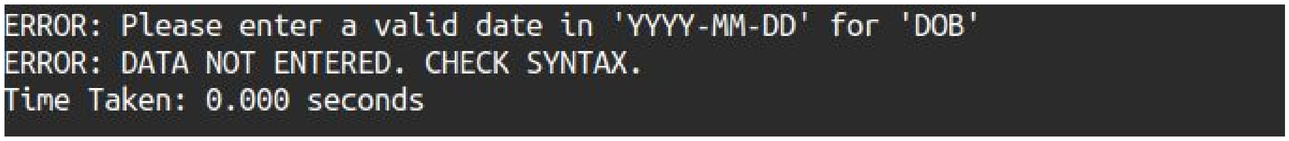


```
ERROR in create: Syntax Error
Time Taken: 0.000 seconds
```

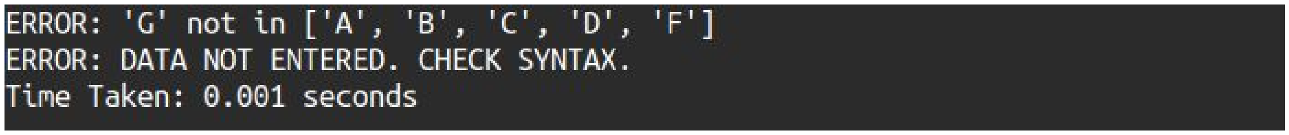
3. ### Data Integrity Constraints:



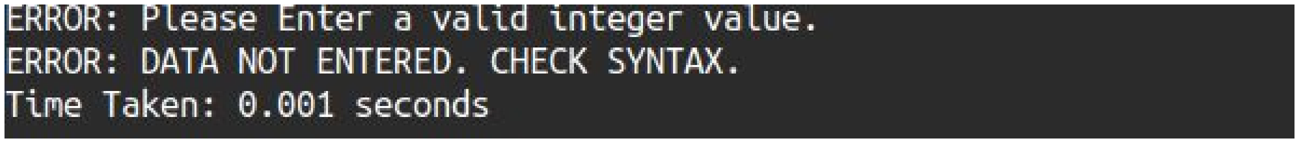
```
ENTERED TUPLE DOES NOT CONTAIN REQUIRED NUMBER OF VALUES.
ERROR: DATA NOT ENTERED.
Time Taken: 0.002 seconds
```



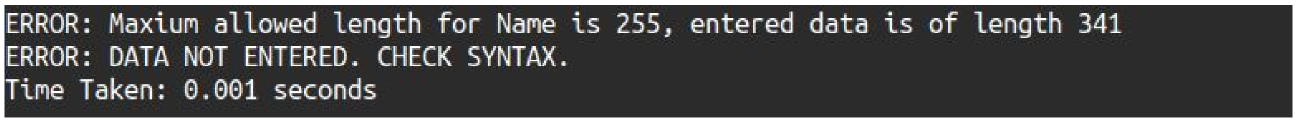
```
ERROR: Please enter a valid date in 'YYYY-MM-DD' for 'DOB'
ERROR: DATA NOT ENTERED. CHECK SYNTAX.
Time Taken: 0.000 seconds
```



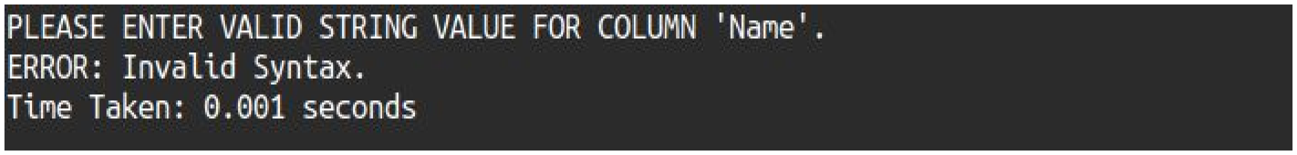
```
ERROR: 'G' not in ['A', 'B', 'C', 'D', 'F']
ERROR: DATA NOT ENTERED. CHECK SYNTAX.
Time Taken: 0.001 seconds
```



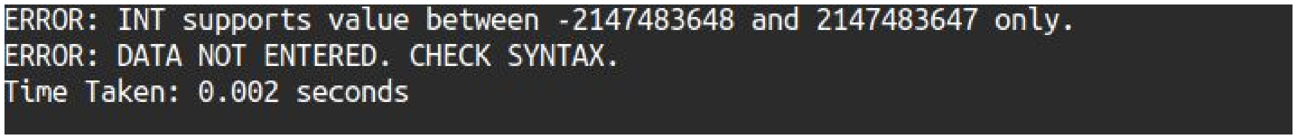
```
ERROR: Please Enter a valid integer value.
ERROR: DATA NOT ENTERED. CHECK SYNTAX.
Time Taken: 0.001 seconds
```



```
ERROR: Maxium allowed length for Name is 255, entered data is of length 341
ERROR: DATA NOT ENTERED. CHECK SYNTAX.
Time Taken: 0.001 seconds
```



```
PLEASE ENTER VALID STRING VALUE FOR COLUMN 'Name'.
ERROR: Invalid Syntax.
Time Taken: 0.001 seconds
```



```
ERROR: INT supports value between -2147483648 and 2147483647 only.
ERROR: DATA NOT ENTERED. CHECK SYNTAX.
Time Taken: 0.002 seconds
```

- Sample Table ‘cities’ source: https://people.sc.fsu.edu/~jburkardt/data/csv/csv.html